DEPARTEMENT TOEGEPASTE ECONOMISCHE WETENSCHAPPEN

ONDERZOEKSRAPPORT NR 9525

Restructuring and Simplifying Rule Bases

by

Jan VANTHIENEN

Elke DRIES

Geert WETS

Katholieke Universiteit Leuven

Naamsestraat 69, B - 3000 Leuven

The decision rules for the problem situation, from which the DT in figure 1 is constructed automatically with the PROLOGA tool, are given below. Note that the rules that are specified in the knowledge acquisition process may be redundant, such as they are specified by the human expert. An optimal rule set is the intended result. Possible contradictions and incompletenesses will be reported by PROLOGA, while the DTs are filled.

- Drastic-restriction **if** number-of-cars >= 150 **and** alternative-route = Y;
- Drastic-restriction **if** (75 <= number-of-cars < 150 **or** number-of-cars >= 150) **and** number-of-accidents >= 20 **and** alternative-route = Y;
- Restriction **if** number-of-cars < 75 **and** number-of-accidents >= 20;
- Restriction **if** 75 <= number-of-cars < 150 **and** 10 <= number-of-accidents < 20 **and** alternative-route = Y;
- Restriction **if** 75 <= number-of-cars < 150 **and** number-of-accidents >= 20 **and** alternative-route = N;
- Restriction **if** number-of-cars >= 150 **and** alternative route = N;
- No-restriction **if** number-of-cars < 75 **and** (number-of-accidents < 10 **or** 10 <= number-of-accidents < 20);
- No-restriction **if** (number-of-cars < 75 **or** 75 <= number-of-cars < 150) **and** number-of-accidents < 10;
- No-restriction **if** (number-of-cars < 75 **or** 75 <= number-of-cars < 150) **and** (number-of-accidents < 10 **or** 10 <= number-of-accidents < 20) **and** alternative-route = N;

	<75		75-<150					>=150	
1. Cars	<75		75-<150					>=150	
2. Accidents	<10 OR 10-<20	>=20	<10	10-<20		>=20		-	
3. Alternative route	-	-	-	Y	N	Y	N	Y	N
1. Drastic restriction	-	-	-	-	-	X	-	X	-
2. Restriction	-	X	-	X	-	-	X	-	X
3. No restriction	X	-	X	-	X	-	-	-	-

Figure 1: Example of a contracted DT

Applying the minimal rule generation algorithm results in the following rule set:

- Drastic-restriction **if** alternative-route = Y **and** ((75 <= number-of-cars < 150 **and** number-of-accidents >= 20) **or** number-of-cars >= 150);
- Restriction **if** (number-of-accidents >= 20 **and** (number-of-cars < 75 **or** alternative-route = N)) **or** (number-of-cars >= 150 **and** alternative-route = N) **or** (75 <= number-of-cars < 150 **and** 10 < number-of-accidents <= 20 **and** alternative-route = Y);
- No-restriction **if** (number-of-cars < 75 **and** (number-of-accidents < 10 **or** 10 <= number-of-accidents < 20)) **or** (75 <= number-of-cars < 150) **and** ((10 < number-of-accidents < 20 **and** alternative-route=N) **or** number-of-accidents < 10));

Large scale problems are modeled in PROLOGA by means of a hierarchy of DTs. Transformation to the minimal rule representation form simply requires the transformation of each table in the hierarchy. Due to the modular structure, changes in the rule base only affect one table.

3. Links with previous research

The problem of transforming DTs into minimal action based decision rules, is related to minimization problems in other research domains.

In the 1950s, the early years of digital design, logic gates were very expensive and the simplification of boolean functions became then an active area of research. Pioneering work was done by Quine and McCluskey, whose techniques gained widespread attention [6] [7]. Their methods involve two major steps (cf. infra): 1. generation of all prime implicants and 2. extraction of a minimum prime cover. Various heuristic approaches to the problem were developed later on for reasons of efficiency. Two categories of algorithms can be distinguished here. One category follows the classical logic minimization techniques, first generating all prime implicants, but, instead of generating a minimum cover, a near minimum cover is selected heuristically. A second category of algorithms tries to simultaneously identify and select implicants for the cover.

In the 1970s, also the DT community started to show interest in logic minimization algorithms. These algorithms were used to minimize the number of rules in a decision grid chart, an intermediate knowledge representation form in constructing DTs. A preliminary minimization of the number of decision rules in the grid chart simplified the time-consuming conversion process into a contracted DT. Strunz applied the Quine-McCluskey method to obtain a minimal decision grid chart [13], while Maes extended the iterative consensus method of McCluskey to make it applicable to extended-entry rules [4].

It will be clear from the following that, although the focus of our research now is completely different, the techniques used to solve the above mentioned minimization problems have been of great use.

4. Minimal rule generation procedure

In the following, the procedure to generate minimal rules from DTs is presented. The different steps will be illustrated by applying them on the DT shown in figure 2. For the sake of conciseness, condition and action subjects are represented by a number, while condition states are represented by a letter.

1	a											b									c			
2	a						b					a						b			a		b	
3	a		b		c		a		b	c		a		b		c		a		b, c	–		–	
4	a	b	a	b	a	b	a	b	–	a	b	a	b	a	b	a	b	a	b	–	a	b	a	b
1	.	×	.	×	.	×	×	×	.	.	.	×	×	.	×	.	×	×	.	.	.	×	×	.
2	.	×	×	×	.	×	.	×	×	.	×	.	×	.	.	×	×	.	×	.	.	.	.	.

Figure 2: Example DT

An action based rule translation of a DT generates one rule for each relevant action (non-empty row) in the DT. With every action mark (x) in a DT, a conjunction of condition states is associated (the condition states of the concerning decision column). A conjunction of condition states that implicates a certain action will be called an *implicant* of that action. The premise of a rule that describes the entire application field of a certain action can be obtained from the DT in a straightforward way: it can be written as a disjunction of the implicants corresponding with the action marks of that action. Figure 3 shows the action based decision rules for the DT in figure 2 when applying this method. As can be seen, the right hand sides of the rules tend to be very complex.

1 **if** $(1a \wedge 2a \wedge 3a \wedge 4b) \vee (1a \wedge 2a \wedge 3b \wedge 4b) \vee (1a \wedge 2a \wedge 3c \wedge 4b)$
$\vee (1a \wedge 2b \wedge 3a \wedge 4a) \vee (1a \wedge 2b \wedge 3a \wedge 4b) \vee (1b \wedge 2a \wedge 3a \wedge 4a)$
$\vee (1b \wedge 2a \wedge 3a \wedge 4b) \vee (1b \wedge 2a \wedge 3b \wedge 4b) \vee (1b \wedge 2a \wedge 3c \wedge 4b)$
$\vee (1b \wedge 2b \wedge 3a \wedge 4a) \vee (1c \wedge 2a \wedge 4b) \vee (1c \wedge 2b \wedge 4a)$

2 **if** $(1a \wedge 2a \wedge 3a \wedge 4b) \vee (1a \wedge 2a \wedge 3b \wedge 4a) \vee (1a \wedge 2a \wedge 3b \wedge 4b)$
$\vee (1a \wedge 2a \wedge 3c \wedge 4b) \vee (1a \wedge 2b \wedge 3a \wedge 4b) \vee (1a \wedge 2b \wedge 3b)$
$\vee (1a \wedge 2b \wedge 3c \wedge 4b) \vee (1b \wedge 2a \wedge 3a \wedge 4b) \vee (1b \wedge 2a \wedge 3c \wedge 4a)$
$\vee (1b \wedge 2a \wedge 3c \wedge 4b) \vee (1b \wedge 2b \wedge 3a \wedge 4b)$

Figure 3: Action based decision rules

The procedure presented in this section consists of three successive steps to simplify these decision rules[1].

4.1. First simplification: contraction of the DT for each action

A first way to obtain less complex action rules from a DT, is by contracting the DT for each action separately and to apply the given method on these tables (one for each action). The more actions a DT contains, the more different action configurations are possible. As only columns with the same action configuration can be combined during the contraction process, it is clear that the possibility to contract decision columns increases heavily by considering only one of the actions during this process. The condition entries of the resulting DTs will contain more irrelevant condition states than the condition entries of the original DT, which results in an action rule with both a decreased number of implicants and less condition states in the implicants. Two types of contraction are possible, dependent on the fact whether the condition order in the DT is maintained or changed. Details about the contraction algorithms can be found in Vanthienen [16].

4.1.1. Table contraction with fixed condition order

In this case the number of columns in the DT is minimized for the given condition order. As can be seen from the DTs resulting from this contraction process and the corresponding decision rules, the reduction of the complexity of the action rules is

[1] It should be noted that the approach can also be applied to rule bases which rules have multiple actions, by first splitting those rules (e.g. $a1 \wedge a2 \leftarrow c1 \wedge c2 \wedge c3 \Leftrightarrow a1 \leftarrow c1 \wedge c2 \wedge c3$ *and* $a2 \leftarrow c1 \wedge c2 \wedge c3$) and then combining the premises of the rules governing the same action.

striking. For DTs with several actions, the simplification will be even more significant.

Action 1:

1. 1	a				b						c			
2. 2	a		b		a			b			a		b	
3. 3	–		a	b, c	a	b, c		a		b, c	–		–	
4. 4	a	b	–	–	–	a	b	a	b	–	a	b	a	b
1. 1	.	×	×	.	×	.	×	×	.	.	.	×	×	.

1 **if** (1a ∧ 2a ∧ 4b) ∨ (1a ∧ 2b ∧ 3a) ∨ (1b ∧ 2a ∧ 3a) ∨ (1b ∧ 2a ∧ (3b ∨ 3c) ∧ 4b) ∨ (1b ∧ 2b ∧ 3a ∧ 4a) ∨ (1c ∧ 2a ∧ 4b) ∨ (1c ∧ 2b ∧ 4a)

Action 2:

1. 1	a					b							c
2. 2	–					a				b			–
3. 3	a		b	c		a		b	c	a		b, c	–
4. 4	a	b	–	a	b	a	b	–	–	a	b	–	–
1. 2	.	×	×	.	×	.	×	.	×	.	×	.	.

2 **if** (1a ∧ 3a ∧ 4b) ∨ (1a ∧ 3b) ∨ (1a ∧ 3c ∧ 4b) ∨ (1b ∧ 2a ∧ 3a ∧ 4b) ∨ (1b ∧ 2a ∧ 3c) ∨ (1b ∧ 2b ∧ 3a ∧ 4b)

4.1.2. Table contraction with change of the condition order

In this case the condition order is determined which results in the minimum number of contracted columns. The condition order is the same for all columns in the DT. Here as well, the optimization process is done for each action separately. Note that the resulting action rules are more compact than those that were derived from the DTs in the previous section.

Action 1:

1. 2	a					b					
2. 4	a				b	a			b		
3. 1	a	b		c	–	a, b		c	a		b, c
4. 3	–	a	b, c	–	–	a	b, c	–	a	b, c	–
1. 1	.	×	.	.	×	×	.	×	×	.	.

1 **if** (2a ∧ 4a ∧ 1b ∧ 3a) ∨ (2a ∧ 4b) ∨ (2b ∧ 4a ∧ (1a ∨ 1b) ∧ 3a) ∨ (2b ∧ 4a ∧ 1c) ∨ (2b ∧ 4b ∧ 1a ∧ 3a)

Action 2:

1. 3	a			b		c				
2. 1	a, b		c	a	b, c	a		b		c
3. 2	–		–	–	–	–		a	b	–
4. 4	a	b	–	–	–	a	b	–	–	–
1. 2	.	×	.	×	.	.	×	×	.	.

2 **if** (3a ∧ (1a ∨ 1b) ∧ 4b) ∨ (3b ∧ 1a) ∨ (3c ∧ 1a ∧ 4b) ∨ (3c ∧ 1b ∧ 2a)

4.2. *Second simplification: minimization of the number of implicants in the premise*

The number of implicants that appear in each decision rule can be minimized by applying the algorithm of Maes for minimizing decision grid charts (DGCs) [4]. A DGC is a multiple hit table which columns relate to one action. In the seventies, it was used as an intermediate knowledge representation form in constructing DTs. The algorithm of Maes is a two step algorithm that minimizes the number of columns in a DGC. The algorithm is based on the iterative consensus method of McCluskey for simplifying switching functions [7]. Maes extended the consensus concept in order to make the method applicable to extended entry rules. We implemented the algorithm and used it in a different context.

We will use a tabular notation to represent implicants of an action. Figure 4 shows the implicant tables (ITs) that can be derived from the DTs in the previous section (4.1.2.). Each column in the table corresponds with an implicant of the concerning action.

A1						
1	b	-	a	b	c	a
2	a	a	b	b	b	b
3	a	-	a	a	-	a
4	a	b	a	a	a	b
I	1	2	3	4	5	6

A2					
1	a	b	a	a	b
2	-	-	-	-	a
3	a	a	b	c	c
4	b	b	-	b	-
I	1	2	3	4	5

Figure 4: The ITs for A_1 and A_2[2]

4.2.1. Construction of the complete sum

This is the first step of the algorithm. The result of it is an IT which contains all the prime implicants of the concerning action. It is obtained by applying the iterative consensus method. The necessary definitions, modified for our purpose, are given below.

Definitions

Let A be an action of a DT.

An implicant α of A is said to *include* another implicant β of A if for each combination of condition states for which β is true, α is true.

An implicant α of A is a *prime implicant* of A if there exists no implicant $\beta \neq \alpha$ of A such that β includes α.

The *complete sum* of A is the IT consisting of the prime implicants of A.

[2] The numbers in the left column represent the condition names and not their order of appearance in the DT from which the IT is derived.

N implicants $\alpha_1, \alpha_2, \ldots, \alpha_N$ of an IT have a *consensus*
if and only if

1. $\exists C_i : (CT_i = \{S_{ik}\}$ has N elements) and ($\forall S_{ik} \in CT_i : S_{ik}$ is contained in exact one of the implicants $\alpha_1, \alpha_2, \ldots, \alpha_N$),

and

2. $\forall C_j, j \neq i$: the implicants from $\{\alpha_1, \alpha_2, \ldots, \alpha_N\}$ that have no don't care as condition state for C_j all have the same condition state for C_j.

C_i is called the *discretionary condition*.

From N implicants $\alpha_1, \alpha_2, \ldots, \alpha_N$ of an IT having a consensus with C_i as the discretionary condition, a *consensus implicant* is formed by filling in:

1. for C_i: a don't care,
2. for C_j with $j \neq i$:
   - a don't care if all implicants $\alpha_1, \alpha_2, \ldots, \alpha_N$ have a don't care as condition state for C_j,
   - the unique condition state appearing for C_j in either of the implicants $\alpha_1, \alpha_2, \ldots, \alpha_N$ in the other case.

The algorithm

The complete sum can be obtained by the successive addition of consensus implicants and the removal of implicants included in others. The following process is repeatedly carried out:

For each condition C_i **do**
  While a consensus exists between N implicants of the IT with C_i as the discretionary condition **do**
    Begin
      Construct the consensus implicant and save it if it is not included in an implicant of the IT or an existing consensus implicant;
      Omit all implicants from the IT and all consensus implicants that are included in the new consensus implicant
    End;
Add the consensus implicants to the IT;

The iteration of the process terminates when execution does not result in the addition of at least one consensus implicant.

Illustration

As an illustration, the complete sums for the ITs of figure 4 are constructed.

Action 1:

- Condition 1 as discretionary condition:
  The consensus between implicants 3, 4 and 5 produces consensus implicant 7. Implicants 3 and 4 are included in the consensus implicant and are deleted from the table.
- Condition 2 as discretionary condition:
  The consensus between implicants 2 and 6 produces consensus implicant 8. Implicant 6 disappears.
- Condition 3 as discretionary condition:
  No consensus possible.

- Condition 4 as discretionary condition:
  The consensus between implicants 1 and 2 produces consensus implicant 9.
  Implicant 1 disappears.

Figure 5 shows the resulting IT. The marked columns are deleted from it.

A1									
1	b	-	a	b	c	a	-	a	b
2	a	a	b	b	b	b	b	-	a
3	a	-	a	a	-	a	a	a	a
4	a	b	a	a	a	b	a	b	-
I	1	2	3	4	5	6	7	8	9

Figure 5: The IT for A_1 after the first iteration

- Condition 1 as discretionary condition:
  No consensus possible.
- Condition 2 as discretionary condition:
  The consensus between implicants 7 and 9 produces consensus implicant 10.
- Condition 3 as discretionary condition:
  No consensus possible.
- Condition 4 as discretionary condition:
  The consensus between implicants 7 and 8 produces consensus implicant 11.

Figure 6 shows the resulting IT.

A1							
1	-	c	-	a	b	b	a
2	a	b	b	-	a	-	b
3	-	-	a	a	a	a	a
4	b	a	a	b	-	a	-
I	2	5	7	8	9	10	11

Figure 6: The IT for A_1 after the second iteration

The third iteration does not result in the addition of new consensus implicants. The complete sum for action A_1 is the table shown in figure 6.

Action 2:

The result of the computation of the complete sum for action A_2 is:

A2				
1	b	a	b	a
2	-	-	a	-
3	a	b	c	-
4	b	-	-	b
I	2	3	5	6

Figure 7: The complete sum for action A_2

4.2.2. Construction of the minimum sum

This is the second part of the algorithm. The minimal IT is formed by picking the fewest prime implicants as possible from the complete sum. If more than one IT exists having the minimal number of columns, then only the IT with the highest number of don't cares will be taken as the minimum sum. The selection is done by giving cardinal numbers to each prime implicant in the complete sum, one unique number for each extended implicant (i.e. without irrelevant conditions) that is included in the prime implicant itself. Prime implicants whose cardinal numbers all appear as cardinal numbers of other prime implicants are redundant and can be removed from the IT.

Usually, there are several orders in which redundant prime implicants can be eliminated, and some of these orders will result in minimum sums and others may not. In the latter case an *irredundant sum* is obtained. This is a set of implicants that can not be reduced without changing the application area of the action.

In [6] different procedures are presented to select implicants for the minimum sum from the complete sum. However, all these procedures are rather cumbersome and not efficient to implement. For the time being, we have therefore chosen to implement a simplified and more efficient algorithm in the PROLOGA workbench, that eliminates redundant implicants in the order in which they are met. This algorithm does not guarantee a minimal solution in all cases, but experience has shown that in all cases a minimal solution or a reasonable approximation is obtained.

Numbering mechanism

1. Each condition is given a *multiplication factor* F_i:
   F_i is the multiplication factor of condition C_i and is defined as:
   $F_{cnum} = 1$,
   $F_i = F_{i+1}$ * (number of condition states for C_{i+1}) for i = cnum - 1, cnum - 2, ..., 1.

2. Each condition state is given a *weight factor* W_{ij}:
   W_{ij} is the weight factor of state number j of condition i and is defined as $W_{ij} = j - 1$.

3. Each extended implicant is given a *cardinal number* N:
   $$N = \sum_{i=1}^{cnum} (F_i \times W_{ij}).$$

4. The cardinal numbers of a prime implicant are the cardinal numbers of the extended implicants contained in it.

The algorithm

First the cardinal numbers of all the prime implicants in the complete sum are calculated. Then the prime implicants are examined one by one. If all cardinal numbers of a prime implicant appear as cardinal number of another prime implicant, then the implicant is redundant and can be deleted.

For all prime implicants α_i in the complete sum
do calculate the cardinal numbers of the prime implicant;
For all prime implicants α_i in the complete sum **do**
Begin
While (not all prime implicants $\alpha_j \neq \alpha_i$ are treated)
and (not all cardinal numbers of α_i are marked) **do**
mark the cardinal numbers of α_i that appear in α_j;
If all cardinal numbers of α_i are marked **then** delete α_i
End;

Illustration

As an illustration, the minimal sum for A_1 and A_2 is determined.

The multiplication factors and weight factors of the conditions are:

Condition 1:	$F_1 = 12$	$W_{11} = 0$	$W_{12} = 1$	$W_{13} = 2$
Condition 2:	$F_2 = 6$	$W_{21} = 0$	$W_{22} = 1$	
Condition 3:	$F_3 = 2$	$W_{31} = 0$	$W_{32} = 1$	$W_{33} = 2$
Condition 4:	$F_4 = 1$	$W_{41} = 0$	$W_{42} = 1$	

<u>*Action 1:*</u>

The cardinal numbers of implicant 2 of the complete sum for A_1 are calculated as follows (see figure 6):

$(F_1 \times W_{11}) / (F_1 \times W_{12}) / (F_1 \times W_{13}) + (F_2 \times W_{21}) + (F_3 \times W_{31}) / (F_3 \times W_{32}) / (F_3 \times W_{33})$
$+ (F_4 \times W_{42})$
$= (12 \times 0) / (12 \times 1) / (12 \times 2) + (6 \times 0) + (2 \times 0) / (2 \times 1) / (2 \times 2) + (1 \times 1)$
$= 1 / 3 / 5 / 13 / 15 / 17 / 25 / 27 / 29$

The cardinal numbers of the other prime implicants are calculated in a similar way:

I 5 : 30, 32, 34
I 7 : 6, 18, 30
I 8 : 1, 7
I 9 : 12, 13
I 10 : 12, 18
I 11 : 6, 7

Applying the algorithm results in the elimination of implicants 7, 8 and 9. The result is the minimum sum (see figure 8). Notice that the elimination of for instance implicants 8 and 10 would have resulted in an irredundant sum.

A1				
1	-	c	b	a
2	a	b	-	b
3	-	-	a	a
4	b	a	a	-
I	2	5	10	11

Figure 8: The minimum sum for action A_1

1 **if** $(2a \wedge 4b) \vee (1c \wedge 2b \wedge 4a) \vee (1b \wedge 3a \wedge 4a) \vee (1a \wedge 2b \wedge 3a)$

Action 2:

The cardinal numbers of the prime implicants of the complete sum for A_2 are (see figure 7):

I 2 : 13, 19
I 3 : 2, 3, 8, 9
I 5 : 16, 17
I 6 : 1, 3, 5, 7, 9, 11

There are no redundant implicants. In this case, the complete sum is identical with the minimum sum.

2 **if** $(1b \wedge 3a \wedge 4b) \vee (1a \wedge 3b) \vee (1b \wedge 2a \wedge 3c) \vee (1a \wedge 4b)$

4.3. *Third simplification: factorization*

At this point, the number of implicants appearing in each action rule is irreducible. However, using the distributivity theorems, the rules can be further simplified. The following heuristic procedure is proposed:

1. Count for each condition state the number of appearances in the minimum sum.
2. Determine the maximum of these numbers. Let S_{max} be the corresponding condition state.
3. Factorize the decision rule with respect to S_{max} and apply the same procedure on the subset of implicants in which the condition state S_{max} appears, thereby not considering the condition state S_{max} anymore, and on the subset of implicants in which the condition state S_{max} not appears.

The recursion stops when the maximum of the condition state frequencies equals 1, in which case factorization is not possible anymore. If more than one condition state exists, having the maximum number of appearances, the following procedure is applied to determine the factorizing condition state:

1. List for each appearance of the maximum the corresponding condition state S_{ik}.
2. Calculate for each such S_{ik} the rest maximum, this is the maximum condition state frequency in the subset of implicants in which S_{ik} not appears.
3. The factorizing condition state is the condition state S_{ik} with the largest rest maximum. If more than one such condition state exists, the first of them is arbitrarily selected.

Illustration

As an illustration, the factorization procedure will now be executed on the decision rules corresponding to the minimum sums in the previous section.

Action 1:

The maximum condition state frequency is 2, corresponding with condition states 2b, 3a and 4a. All condition states have the same rest maximum being 0. The rule is factorized with respect to condition state 2b:

1 **if** $(2b \wedge ((1a \wedge 3a) \vee (1c \wedge 4a))) \vee (1b \wedge 3a \wedge 4a) \vee (2a \wedge 4b)$

Action 2:

The maximum condition state frequency is 2, corresponding with condition states 1a, 1b and 4b. Condition states 1a and 1b have the largest rest maximum 2.

Condition state 1a is chosen as the factorizing condition, yielding the following rule for action A_2:

2 **if** $(1a \wedge (3b \vee 4b)) \vee (1b \wedge 2a \wedge 3c) \vee (1b \wedge 3a \wedge 4b)$

The factorizing procedure is now applied to the implicant subset consisting of the implicants 3 and 6, thereby not considering condition state 1a. The maximum condition state frequency is 1, so there is no factorization possible anymore. The implicants 2 and 5 can be further factorized with respect to condition state 1b. This yields the following final result for action rule 2:

2 **if** $(1a \wedge (3b \vee 4b)) \vee (1b \wedge ((2a \wedge 3c) \vee (3a \wedge 4b)))$

Conclusion

The representational capabilities of the decision table make it a valuable tool in knowledge acquisition and verification and validation. The knowledge enclosed in a decision table, can be implemented in several ways. In this paper an algorithm is presented to convert decision tables into a minimal rule representation.
The proposed conversion facility allows automatic optimal rule generation from decision tables and verification and optimization of rule bases and other specifications. It faces the emerging problems of increasing complexity and maintenance of rule bases.

References

[1] Ginsberg, A., Knowledge-Base Reduction: A New Approach to Checking Knowledge Bases for Inconsistency & Redundancy, *Proc. Seventh National Conf. on Artificial Intelligence* (AAAI88), Saint Paul, Minnesota (1988) 585-589.

[2] Hicks, R. C., Minimizing Maintenance Anomalies in Expert Systems, *Information & Management* 28 (1995) 177-184.

[3] Higa, K., Owei, V., An Integrated Rule-Base Reduction System, *Proc. IEEE/ACM Int. Conf. on Developing and Managing Expert System Programs*, Washington, D.C. (1991) 120-128.

[4] Maes, R., On Minimizing Decision Grid Charts, *Angewandte Informatik* (1982) 451-455.

[5] Marathe, H., Ma, T.-K., Liu, C.-C., An Algorithm for Identification of Relations among Rules, *Proc. IEEE International Workshop on Tools for Artificial Intelligence* (TAI89), Fairfax, Virginia (1989) 360-367.

[6] McCluskey, E.J., Minimization of Boolean Functions, *The Bell System Technical Journal* (1956) 1417-1444.

[7] McCluskey, E.J., *Introduction to the Theory of Switching Circuits* (McGraw-Hill, New York, 1965).

[8] Nonfjall, H., Larsen, H. L., Detection of Potential Inconsistencies in Knowledge Bases, *The International Journal of Intelligent Systems* 7 (1992) 81-96.

[9] O'Keefe, R. M., O'Leary, D. E., Expert System Verification and Validation: a Survey and Tutorial, *Artificial Intelligence Review* 7 (1993) 3-42.

[10] Polat, F., UVT: A Unification-Based Tool for Knowledge Base Verification, *IEEE Expert* (1993) 69-75.

[11] Preece, A. D., Shinghal, R., Foundation and Application of Knowledge Base Verification, *International Journal of Intelligent Systems.* 9 (1994) 683-701.

[12] Santos-Gomez, L., Darnell, M. J., Empirical Evaluation of Decision Tables for Constructing and Comprehending Expert System Rules, *Knowledge Acquisition* 4 (1992) 427-444.

[13] Strunz, H., Grundlagen und Anwendungsmöglichkeiten der Entscheidungstabellentechnik bei der Gestaltung rechnergestützter Informationssysteme, Doctoral Dissertation, University of Köln (West Germany), 1975.

[14] Tanaka, M., Aoyama, N., Sugiura, A., Koseki, Y., Integration of Multiple Knowledge Representation for Classification Problems, *Proc. Fifth Int. Conf. on Tools with Artificial Intelligence* (ICTAI93), Boston, Mass. (1993) 448-449.

[15] Vanhaelemeesch, T., *Ontwerp van Programmatuur voor het Genereren van Actiegerichte Beslissingsregels in PROLOGA*, Dissertation, K.U.Leuven, Dept. of Applied Economic Sciences, 1993.

[16] Vanthienen, J., *Automatiseringsaspecten van de Specificatie, Constructie en Manipulatie van Beslissingstabellen*, Doctoral Dissertation, K.U.Leuven, Dept. of Applied Economic Sciences, 1986.

[17] Vanthienen, J., Knowledge Acquisition and Validation Using a Decision Table Engineering Workbench, *Proc. World Congress on Expert Systems*, Orlando, Florida (1991) 1861-1868.

[18] Vanthienen, J., Dries, E., Illustration of a Decision Table Tool for Specifying and Implementing Knowledge Based Systems, *International Journal on Artificial Intelligence Tools* 3 (1994) 267-288.

[19] Vanthienen, J., Dries, E., Decision Tables: Refining the Concept and a Proposed Standard, to appear in *Communications of the ACM.*

[20] Vanthienen, J., Wets, G., From Decision Tables to Expert System Shells, *Data & Knowledge Engineering* 13 (1994) 265-282.

Contents